Journal of Computational and Applied Mathematics 236 (2012) 3101–3110

Contents lists available at SciVerse ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

A modified conjugate gradient algorithm with cyclic Barzilai–Borwein steplength for unconstrained optimization

Yunhai Xiao *, Huina Song, Zhiguo Wang

Institute of Applied Mathematics, College of Mathematics and Information Science, Henan University, Kaifeng 475000, China

ARTICLE INFO

Article history:
Received 18 January 2011
Received in revised form 14 June 2011

Keywords:
Conjugate gradient method
CG_DESCENT
Barzilai–Borwein steplength
Wolfe condition
CUTEr library

ABSTRACT

For solving large-scale unconstrained minimization problems, the nonlinear conjugate gradient method is welcome due to its simplicity, low storage, efficiency and nice convergence properties. Among all the methods in the framework, the conjugate gradient descent algorithm — CG_DESCENT is very popular, in which the generated directions descend automatically, and this nice property is independent of any line search used. In this paper, we generalize CG_DESCENT with two Barzilai–Borwein steplength reused cyclically. We show that the resulting algorithm owns attractive sufficient descent property and converges globally under some mild conditions. We test the proposed algorithm by using a large set of unconstrained problems with high dimensions in CUTEr library. The numerical comparisons with the state-of-the-art algorithm CG_DESCENT illustrate that the proposed method is effective, competitive, and promising.

1. Introduction

In this paper, we consider

$$\min f(x), \quad x \in \mathbb{R}^n, \tag{1.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function, whose gradient at point x_k is $g(x_k)$, or g_k, for the sake of simplicity; n is the number of variables, which is assumed to be large.

Numerous algorithms have been proposed, analyzed, and tested in the past decades to solve (1.1), for instance, limited memory quasi-Newton method, conjugate gradient method and Barzilai–Borwein gradient method [1]. Among them, the conjugate gradient method has received a great deal of attention, especially when n is large. The iterative form of this method is

$$x_{k+1} = x_k + \alpha_k d_k, \tag{1.2}$$

where α_k is a steplength, and d_k is a search direction which is defined by

$$d_k = \begin{cases} -g_0, & \text{if } k = 0, \\ -g_k + \beta_k d_k, & \text{if } k \geq 1, \end{cases} \tag{1.3}$$

where β_k is a scalar, and different choices mean different conjugate gradient methods. The best-known formulas for β_k are the Fletcher–Reeves (FR), Polak–Ribiere–Polyak (PRP), Hestenes–Stiefel (HS), and Dai–Yuan (DY) methods [2–5] which are given by

$$\beta_k^{\mathrm{FR}} = \frac{\|g_k\|^2}{\|g_{k-1}\|^2}, \qquad \beta_k^{\mathrm{PRP}} = \frac{g_k^\top y_{k-1}}{\|g_{k-1}\|^2}, \qquad \beta_k^{\mathrm{HS}} = \frac{g_k^\top y_{k-1}}{d_{k-1}^\top y_{k-1}}, \quad \text{and} \quad \beta_k^{\mathrm{DY}} = \frac{\|g_k\|^2}{d_{k-1}^\top y_{k-1}},$$

* Correspondence to: Institute of Applied Mathematics, Henan University, Kaifeng 475000, China.
E-mail addresses: yhxiao@henu.edu.cn (Y. Xiao), comsonghuinashn@sina.com (H. Song), wangzg@henu.edu.cn (Z. Wang).

0377-0427/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2012.01.032

where $y_{k-1} = g_k - g_{k-1}$ and symbol $\|\cdot\|$ is the Euclidean norm of a vector. The corresponding methods are generally specified as FR, PRP, HS, and DY methods. When f is strictly convex quadratic and the steplength α_k is the exact one-dimensional minimizer, these methods are equivalent [1]. However, in the non-convex case, their theoretical properties and practical performance may be significantly different.

In practical computation, the PRP method is generally believed to be the most efficient. However, it may cycle infinitely without approaching any stationary points in non-convex minimization [6]. Therefore, much effort has been investigated to create new β_k, which not only possesses global convergence for general functions but is superior to original method [7]. The non-negative setting $\beta_k = \max\{\beta_k^{\mathrm{PR}}, 0\}$ ensures the descent property but it is not always efficient [8]. Another significant work on the convergence of PRP method due to Grippo and Lucidi [9], in which an efficient line search is developed to ensure that the objective function reduces greatly. In the past few years, two classes of method have been received much attention in the literature. This first class falls into the new secant condition based algorithms [10]. The earliest method in this class due to Dai and Liao [11], in which the new formula for β_k is

$$\beta_k^{\mathrm{DL}} = \frac{g_k^\top (y_{k-1} - t s_{k-1})}{d_{k-1}^\top y_{k-1}}, \tag{1.4}$$

where t is a positive scalar and $s_{k-1} = x_k - x_{k-1}$. The second one is the sufficient descent method which means all the generated directions descend automatically without requiring line search. The famous one in this class is CG_DESCENT [12], where

$$\beta_k^{\mathrm{ZH}} = \frac{1}{d_{k-1}^\top y_{k-1}} \left(y_{k-1} - 2 \frac{\|y_{k-1}\|^2}{d_{k-1}^\top y_{k-1}} d_{k-1} \right)^\top g_k. \tag{1.5}$$

The definition of β_k^{ZH} guarantees that $g_k^\top d_k \le -\frac{7}{8}\|g_k\|^2$ – the famous sufficient descent condition which is important for several iterative algorithms convergence globally [13,14].

The main contributions of this paper are to propose a new formula for β_k on the basis of the definition of β_k^{ZH} in (1.5). Our motivation mainly comes from the well-known Barzilai–Borwein (BB) gradient method [15], which is essentially a steepest descent method with a suitable steplength along the negative gradient direction. There are two typical choices in this method. In this paper, we note that one of the choices for t in (1.4) can reduce to β_k^{ZH}, while the other one reduces a new formula for β_k. To get better performance, we choose the BB steplength cyclically with consecutive iterations. The proposed method owns the nice property, that is, all the generated directions always descend. For strongly convex and general non-convex functions, we respectively show that the resulting conjugate gradient method converges globally by using standard Wolfe conditions. We test our method on a large set of unconstrained optimization in CUTEr library [16], and the results illustrate that the proposed algorithm performs better than the state-of-the-art algorithm CG_DESCENT.

The remainder of this paper is organized as follows. In Section 2, we list the steps of our algorithm. In Section 3, we prove its global convergence. In Section 4, we test the performance of the proposed algorithm and compare it with CG_DESCENT.

2. Algorithm

This section is devoted to reviewing the BB gradient method briefly, and list the steps of our algorithm. The BB gradient method is to solve smooth unconstrained minimization problems (1.1) with the iterative form

$$x_{k+1} = x_k - \lambda_k^{-1} g_k$$

where λ_k is named BB steplength with two choices

$$\lambda_k^1 = \frac{s_{k-1}^\top y_{k-1}}{\|s_{k-1}\|^2},$$

and

$$\lambda_k^2 = \frac{\|y_{k-1}\|^2}{s_{k-1}^\top y_{k-1}}.$$

Moreover, some practical experiments have shown that choice λ_k^1 is promising. Earlier theoretical work have shown that BB gradient method converges globally for quadric convex minimization [15,17,18] and for general non-convex unconstrained minimization [19] and bound-constrained minimization [20,21] by incorporating a nonmonotone line search. Moreover, recent results have shown that if the BB steplength is reused for fixed number consecutive iterations, the performance will be greatly improved and superlinear local convergence rate will be achieved [22,23].

Now we reconsider the conjugate gradient method with the iterative form (1.2)–(1.3). Particularly, we only turn our attention to the conjugate gradient formula for β_k^{DL}. Clearly, when the parameter t is taken as $\xi\lambda_k^1$ and $\xi\lambda_k^2$, then (1.4) reduces respectively to

$$\beta_k^1 = \frac{y_{k-1}^\top g_k}{d_{k-1}^\top y_{k-1}} - \xi \frac{d_{k-1}^\top g_k}{\|d_{k-1}\|^2}, \tag{2.1}$$

and

$$\beta_k^2 = \frac{y_{k-1}^\top g_k}{d_{k-1}^\top y_{k-1}} - \xi \frac{\|y_{k-1}\|^2 g_k^\top d_{k-1}}{(d_{k-1}^\top y_{k-1})^2}. \tag{2.2}$$

where $\xi > 0$ is a positive scalar. An interesting fact is that the formula β_k^2 is exactly the β_k^{ZH} in (1.5) in the case of $\xi = 2$. To improve the performance, like [22,23], we also use a cyclic technique, that is, for pre-fixed positive integer M, we choose our new conjugate gradient formula for β_k as

$$\beta_k = \begin{cases} \beta_k^1, & \text{if } \mathrm{mod}(k, M) = 0, \\ \beta_k^2, & \text{otherwise.} \end{cases} \tag{2.3}$$

As we all know that the main advantage of $\beta_k^2 (= \beta_k^{\mathrm{ZH}}$ with $\xi = 2)$ is that the corresponding direction satisfies the sufficient descent condition $d_k^\top g_k \le -\frac{7}{8}\|g_k\|^2$. Fortunately, under appropriate conditions, we can get a similar conclusion.

Lemma 2.1. *If $g_{k-1}^\top d_{k-1} \neq 0$ and $\mathrm{mod}(k, M) = 0$. Then*

$$g_k^\top d_k \le -\frac{7}{8}\|g_k\|^2, \tag{2.4}$$

provided that

$$\xi \ge \frac{2\|y_{k-1}\|\ \|d_{k-1}\|}{|d_{k-1}^\top y_{k-1}|}. \tag{2.5}$$

Proof. When $\mathrm{mod}(k, M) = 0$,

$$d_k = -g_k + \left(\frac{y_{k-1}^\top g_k}{d_{k-1}^\top y_{k-1}} - \xi \frac{d_{k-1}^\top g_k}{\|d_{k-1}\|^2} \right) d_{k-1}.$$

Multiplying both sides by g_k, yields

$$\begin{aligned} g_k^\top d_k &= -\|g_k\|^2 + \left(\frac{y_{k-1}^\top g_k}{d_{k-1}^\top y_{k-1}} - \xi \frac{d_{k-1}^\top g_k}{\|d_{k-1}\|^2} \right) d_{k-1}^\top g_k \\ &= \frac{y_{k-1}^\top g_k g_k^\top d_{k-1} d_{k-1}^\top y_{k-1} \|d_{k-1}\|^2 - \xi (g_k^\top d_{k-1})^2 (d_{k-1}^\top y_{k-1})^2 - \|g_k\|^2 (d_{k-1}^\top y_{k-1})^2 \|d_{k-1}\|^2}{(d_{k-1}^\top y_{k-1})^2 \|d_{k-1}\|^2}. \end{aligned} \tag{2.6}$$

Applying the inequality

$$u^T v \le \frac{1}{2}(\|u\|^2 + \|v\|^2)$$

to the first term in (2.6) with

$$u = \frac{1}{2}(d_{k-1}^\top y_{k-1})\|d_{k-1}\| g_k, \quad \text{and} \quad v = 2 g_k^\top d_{k-1} \|d_{k-1}\| y_{k-1},$$

we have

$$g_k^\top d_k \le -\frac{7}{8}\|g_k\|^2 + \frac{(d_{k-1}^\top g_k)^2 [2\|y_{k-1}\|^2 \|d_{k-1}\|^2 - \xi (d_{k-1}^\top y_{k-1})^2]}{(d_{k-1}^\top y_{k-1})^2 \|d_{k-1}\|^2}.$$

Hence, (2.4) holds for a proper ξ value. □

For the second choice of β_k^2, we have the following similar result.

Lemma 2.2. *If $g_{k-1}^\top d_{k-1} \neq 0$ and $\mathrm{mod}(k, M) \neq 0$. If $\xi \ge 2$, then (2.4) holds.*

Proof. By [12, Theorem 1.1], it is not difficult to deduce that

$$g_k^\top d_k \le -\left(1 - \frac{1}{4\xi}\right)\|g_k\|^2.$$

Hence the lemma's conclusion follows when $\xi \ge 2$. □

Lemmas 2.1 and 2.2 illustrate that sufficient descent condition always holds if ξ is properly chosen.

Theorem 2.1. *Let d_k be defined by (1.3) and (2.3) with positive integer M. If*

$$\xi \geq \max\left\{2, \frac{2\|y_{k-1}\|\|d_{k-1}\|}{|d_{k-1}^{\top} y_{k-1}|}\right\}, \tag{2.7}$$

then (2.4) holds.

Now we formally state the steps of the Conjugate Gradient with Cyclic BB steplength (CGCBB) to solve problem (1.1) as follows.

Algorithm 2 (MPRP)

Step 0 Given x_0, positive constants M and $0 < \rho < \sigma < 1$. Set $k := 0$;
Step 1. Stop if $\|\nabla f(x_k)\| = 0$; Otherwise, continue;
Step 2. Find ξ satisfies (2.7);
Step 3. Determine d_k by (1.3) and (2.2);
Step 4. Find α_k satisfies the Wolfe conditions

$$\begin{cases} f(x_k + \alpha_k d_k) \leq f(x_k) + \rho\alpha_k g_k^{\top} d_k, \\ g(x_k + \alpha_k d_k) \geq \sigma g_k^{\top} d_k. \end{cases} \tag{2.8}$$

Set $x_{k+1} := x_k + \alpha_k d_k$.
Step 5. Set $k := k + 1$. Go to Step 1.

3. Convergence analysis

In this section, we will prove the global convergence of our method. Throughout this section we assume that $g_k \neq 0$ for all $k > 0$ unless a stationary point is found. Although the search directions always descend, we will constrain the choice of α_k to satisfying the Wolf conditions which ensures that the algorithm convergence globally.

We make the following basic assumptions on the objective function.

Assumption 3.1. The level set $\mathcal{L} = \{x | f(x) \leq f(x_0)\}$ is bounded.

Assumption 3.2. In some neighborhood $\mathcal{N}$ of $\mathcal{L}$, f is continuously differentiable, and its gradient is Lipschitz continuous, that is, there exists a constant $L > 0$ such that

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \quad \forall x, y \in \mathcal{N}. \tag{3.1}$$

The above assumptions on f show that its gradient is bounded, namely, there exists a constant $\bar{\gamma}$ such that

$$\|\nabla f(x)\| \leq \bar{\gamma}, \quad \forall x \in \mathcal{L}. \tag{3.2}$$

We now state the Zoutendijk condition [24] without any proof which is important for the convergence of our algorithm.

Lemma 3.1. *Suppose that d_k is a descent direction and the Assumptions 3.1 and 3.2 hold. If the standard Wolf conditions (2.8) is used, then*

$$\sum_{k=0}^{\infty} \frac{(g_k^{\top} d_k)^2}{\|d_k\|^2} < +\infty. \tag{3.3}$$

Lemma 3.2. *Suppose that d_k is a descent direction and the Assumptions 3.1 and 3.2 hold. If the standard Wolf conditions (2.8) is used, then,*

$$\alpha_k \geq \frac{1 - \sigma}{L} \frac{|g_k^{\top} d_k|}{\|d_k\|^2}. \tag{3.4}$$

Proof. Subtracting $g_k^{\top} d_k$ from both sides of the second equation in (2.8), we have

$$(\sigma - 1) g_k^{\top} d_k \leq y_k^{\top} d_k.$$

By Assumption 3.2, it follows that

$$(\sigma - 1) g_k^{\top} d_k \leq \alpha_k L \|d_k\|^2.$$

Since $d_k^{\top} g_k < 0$ and $0 < \sigma < 1$, then we obtain (3.4). □

We now prove global convergence of Algorithm 2.1 with the standard Wolf conditions (2.8) for strongly convex functions.

Theorem 3.1. *Suppose f is a strongly convex function on $\mathcal{L}$, that is, there exists a constant $\mu > 0$ such that*

$$\mu\|x-y\|^2 \le (\nabla f(x) - \nabla f(y))^\top (x-y), \quad \forall x, y \in \mathcal{L}.$$

Let $\{x_k\}$ be generated by Algorithm 2.1. Then either $g_k = 0$ for some k, or

$$\lim_{k\to\infty} g_k = 0.$$

Proof. Assume that $g_k \neq 0$ for all k. Due to the strong convexity,

$$y_k^\top d_k = (g_{k+1} - g_k)^\top d_k \ge \mu\alpha_k\|d_k\|^2. \tag{3.5}$$

The assumption $g_k \neq 0$ implies that $d_k \neq 0$, then $y_k^\top d_k > 0$ by (3.5). By adding both sides of the first implementation (2.8), we have

$$\sum_{k=0}^{\infty} \alpha_k g_k^\top d_k > -\infty. \tag{3.6}$$

Combining with (2.4) and (3.4), it is easy to verify that

$$\sum_{k=0}^{\infty} \frac{\|g_k\|^4}{\|d_k\|^2} < \infty. \tag{3.7}$$

For $\mathrm{mod}(k, M) \neq 0$, it is shown that [12]

$$|\beta_k| \le \left(\frac{L}{\mu} + \frac{\xi L^2}{\mu^2}\right)\frac{\|g_{k+1}\|}{\|d_k\|}. \tag{3.8}$$

For $\mathrm{mod}(k, M) = 0$, utilizing (3.5) and the fact that $\|y_k\| \le L\alpha_k\|d_k\|$, we have

$$\begin{aligned} |\beta_k| &= \left|\frac{y_k^\top g_{k+1}}{d_k^\top y_k} - \xi\frac{d_k^\top g_{k+1}}{\|d_k\|^2}\right| \\ &\le \frac{\|y_k\|\,\|g_{k+1}\|}{|d_k^\top y_k|} + \xi\frac{\|g_{k+1}\|}{\|d_k\|} \\ &\le \left(\frac{L}{\mu} + \xi\right)\frac{\|g_{k+1}\|}{\|d_k\|}. \end{aligned} \tag{3.9}$$

Denote $r = \max\{\frac{L}{\mu} + \xi, \frac{L}{\mu} + \frac{\xi L^2}{\mu^2}\}$. By above analysis of the bound for β_k and the definition of d_k,we have

$$\|d_k\| \le \|g_k\| + |\beta_k|\,\|d_{k-1}\| \le (1+r)\|g_k\|.$$

Combining this below bound with (3.7) yields

$$\sum_{k=0}^{\infty} \|g_k\|^2 < \infty$$

which implies $\lim_{k\to\infty} g_k = 0$. □

For general non-convex function, we have the similar convergence result.

Theorem 3.2. *Let $\{x_k\}$ be generated by Algorithm 2.1 for non-convex functions. Then we have either $g_k = 0$ or*

$$\liminf_{k\to\infty} g_k = 0. \tag{3.10}$$

Proof. For the sake of contradiction, we suppose that (3.10) does not hold. Then there is a constant $\epsilon > 0$ such that $\|g_k\| \ge \epsilon$ for all $k \ge 0$.

Case I. If $\mathrm{mod}(k, M) \neq 0$. Similarly with the first part proof in [12, Theorem 3.2], it is easy to see that

$$|\beta_k| = |\beta_k^2| \le C_1\|s_{k-1}\|, \tag{3.11}$$

where

$$C_1 = \frac{8}{7}\frac{1}{(1-\sigma)\epsilon^2}\left(L\bar{\gamma} + \xi L^2 D \max\left\{\frac{\sigma}{1-\sigma}, 1\right\}\right)$$

and $D = \max\{\|x-y\| : \forall x, y \in \mathcal{L}\}$.

Case II. If $\mathrm{mod}(k, M) \neq 0$, again from [12, Theorem 3.2], we have

$$y_k^\top d_k \geq (1-\sigma)\frac{7}{8}\epsilon^2.$$

Then,

$$\begin{aligned}
|\beta_k| = |\beta_k^1| &= \left|\frac{y_{k-1}^\top g_k}{d_{k-1}^\top y_{k-1}} - \xi\frac{g_k^\top d_{k-1}}{\|d_{k-1}\|^2}\right| \\
&\leq \frac{1}{|d_{k-1}^\top y_{k-1}|}\left(|y_{k-1}^\top g_k| + \xi\frac{|d_{k-1}^\top g_k||d_{k-1}^\top y_{k-1}|}{\|d_{k-1}\|^2}\right) \\
&\leq \frac{8}{7}\frac{1}{(1-\sigma)\epsilon^2}(\|y_{k-1}\|\,\|g_k\| + \xi\|y_{k-1}\|\,\|g_k\|) \\
&\leq \frac{8}{7}\frac{1}{(1-\sigma)\epsilon^2}(1+\xi)\bar{\gamma}L\|s_{k-1}\| \\
&= C_2\|s_{k-1}\|, \qquad (3.12)
\end{aligned}$$

where

$$C_2 = \frac{8}{7}\frac{1}{(1-\sigma)\epsilon^2}(1+\xi)\bar{\gamma}L.$$

All together, choosing $C = \max\{C_1, C_2\}$, we have

$$|\beta_k| \leq C\|s_k\|,$$

which shows that $|\beta_k|$ is bounded. Furthermore, as it was shown [12, Theorem 3.2] that $\|s_k\|$ and $\|d_k\|$ are also bounded. On the other hand, (3.7) illustrates that

$$\epsilon^4\sum_{k=0}^{\infty}\frac{1}{\|d_k\|^2} \leq \sum_{k=0}^{\infty}\frac{\|g_k\|^4}{\|d_k\|^2} < \infty,$$

which yields a contradiction, because the above inequality shows $\|d_k\|$ trends to infinity. This completes our proof. □

4. Numerical experiments

In this section, we will test the feasibility and effectiveness of the Algorithm 2.1. The algorithm is implemented in Fortran77 code using double precision arithmetic. All runs are performed on a PC (Intel Pentium Dual E2140 1.6 GHz, 512 MB SDRAM) with Linux operations system. Our experiments are performed on a large set of the nonlinear unconstrained problems from the CUTEr [16] library, the second-order derivatives of all the selected problems are available. Since we are interested in large problems, we only consider problems with size at least 50. Altogether, we solves 71 problems.

We perform two classes of numerical experiments. In the first class, we test CGCBB with different parameter values, while in the second class we compare the state-of-the-art code CG_DESCENT [25] to show its efficiency and stability. As in [12], we adjust the upper bound on β_k, that is

$$\beta_k = \max\{\beta_k, \eta_k\}, \quad \text{and} \quad \eta_k = \frac{-1}{\|d_{k-1}\|\min\{\eta, \|g_{k-1}\|\}},$$

where $\eta > 0$ is a constant, and it taken as $\eta = .1$ in each test. To terminate the line search quickly, we choose $\rho = .1$ and $\sigma = .9$ in (2.8). Moreover, we set $\xi = 2$ as a constant for easily running the code. The iterative process of CGCBB is terminated when

$$\|\nabla f(x_k)\|_\infty \leq \max\{10^{-8}, 10^{-12}\|\nabla f(x_k)\|_\infty\}. \qquad (4.1)$$

In the beginning, we test CGCBB with different parameter values of M. Because the computational complexity of each algorithm is same, we compare performances based on number of iterations or number of function evaluations (Nf) and gradient (Ng) evaluations. For CUTEr test set, since the CPU time to evaluate the derivative is about 3 times that to evaluate f itself, we can adopt Nf+3Ng as a measure. The performance of all methods is evaluated using the profile of Dolan and

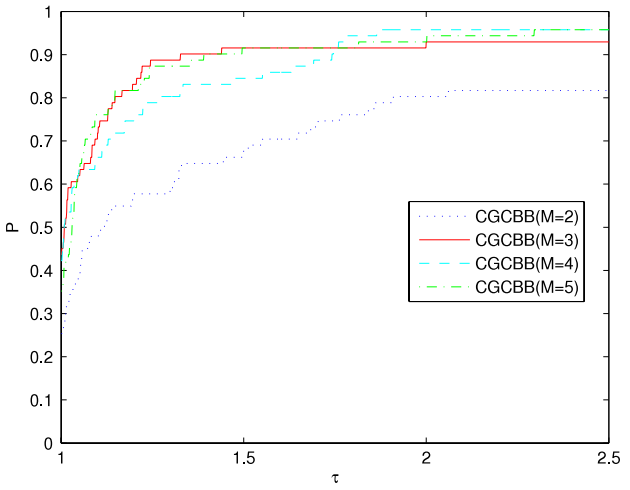

Fig. 1. Performance profiles of CGCBB with different M.

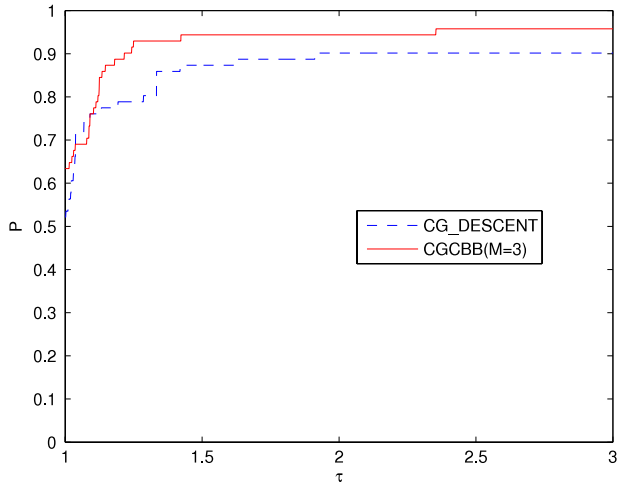

Fig. 2. Performance profiles of CGCBB and CG_DESCENT.

Moré [26]. That is, we plot the fraction P of the test problems that were solved by each choice for cycle length M within a factor τ. Obviously, in a plot of performance profiles, the top curved shape of the algorithm is a winner. Additionally, the right of the plot is a measure of an algorithm's robustness. The performance profile of CGCBB with different M are plotted in Fig. 1.

As can be seen from Fig. 1, the CGCBB algorithm with $M = 2$ is below those with $M = 3, 4, 5$ which almost have identical performance. Fig. 1 also clarifies that cycle length selection parameter M do control the numerical behavior of the algorithm, and bigger value cannot cause better performance. The limited experiments also tell us that $M = 3$ is the best choice.

In the second experiments, we present extensive numerical results of the state-of-the-art algorithm CG_DESCENT. CG_DESCENT is a conjugate gradient descent method for solving large-scale unconstrained optimization problems. A new non-monotone line search used in CG_DESCENT makes this algorithm very efficient. The Fortran code can be obtained from http://www.math.ufl.edu/~hager/. In running of CG_DESCENT, default values are used for all parameters. The algorithm is terminated when the stopping condition (4.1) is met. The results of CG_DESCENT and CGCBB are reported in Tables 1 and 2respectively. Each table reports the name of the test problems (Prob), the dimension (Dim), the number of iterations (Iter), the number of function evaluations (Nf), the number of gradient evaluations (Ng), CPU time in seconds (Time), the final function values (Fv), and the norm of final gradient (Norm).

Observing both of the tables, we see that, in view of Nf+ 3Ng, CGCBB is faster than CG_DESCENT on 28 problems, while CG_DESCENT is superior to CGCBB on 19 problems. Hence, we conclude that CGCBB performs better than CG_DESCENT from the limited experiments. For further and closely examination of the performance of both algorithms, we also plot the performance profile based on Nf+ 3Ng in Fig. 2.

Observing Fig. 2, we conclude that CGCBB is always the top performer for most values of τ, which shows that CGCBB performs better than CG_DESCENT. The left side of this figure gives the percentage of the test problems for which method is

Table 1
Test results for CG_DESCENT.

Prob	Dim	Iter	Nf	Ng	Time	Fv	Norm
ARGLINA	200	1	3	2	0.01	0.20000E+03	0.21133E−06
ARGLINB	200	7	15	15	0.01	0.99625E+02	0.73897E−05
ARGLINC	200	227	803	806	0.69	0.10113E+03	0.71412E−05
ARWHEAD	5000	9	19	12	0.02	0.00000E+00	0.83931E−06
BDQRTIC	5000	1217	2554	1532	3.00	0.20006E+05	0.87039E−05
BROWNAL	200	4	15	13	0.01	0.14731E−08	0.11790E−05
BROYDN7D	5000	1444	2885	1453	5.13	0.19785E+04	0.68854E−05
BRYBND	5000	31	64	34	0.10	0.14292E−10	0.65055E−05
CHAINWOO	4000	272	527	298	0.53	0.45728E+01	0.93722E−05
CHNROSNB	50	245	491	246	0.00	0.11631E−11	0.97612E−05
COSINE	10000	11	31	26	0.07	−0.99990E+04	0.23993E−05
CRAGGLVY	5000	97	188	117	0.34	0.16882E+04	0.97869E−05
CURLY10	10000	64092	79804	112542	251.61	−0.10032E+07	0.93709E−05
CURLY20	10000	100367	120307	180918	610.02	−0.10032E+07	0.98237E−05
DIXMAANA	3000	8	17	9	0.01	0.10000E+01	0.15586E−05
DIXMAANB	3000	9	19	10	0.00	0.10000E+01	0.35546E−06
DIXMAANC	3000	10	21	11	0.01	0.10000E+01	0.70236E−06
DIXMAAND	3000	11	23	12	0.01	0.10000E+01	0.32593E−05
DIXMAANE	3000	194	389	195	0.16	0.10000E+01	0.95540E−05
DIXMAANF	3000	147	295	148	0.13	0.10000E+01	0.96813E−05
DIXMAANG	3000	144	289	145	0.12	0.10000E+01	0.93881E−05
DIXMAANH	3000	140	281	141	0.12	0.10000E+01	0.93579E−05
DIXMAANI	3000	813	1627	814	0.69	0.10000E+01	0.97057E−05
DIXMAANJ	3000	137	275	138	0.12	0.10000E+01	0.96185E−05
DIXMAANL	3000	112	225	113	0.09	0.10000E+01	0.99136E−05
DIXON3DQ	10000	10000	20001	10002	15.09	0.15940E−11	0.54462E−06
DQDRTIC	5000	7	15	8	0.02	0.41826E−14	0.18957E−06
DQRTIC	5000	32	65	33	0.02	0.28560E−04	0.30250E−05
EDENSCH	2000	29	56	33	0.02	0.12003E+05	0.74432E−05
ENGVAL1	5000	23	45	32	0.05	0.55487E+04	0.53164E−05
ERRINROS	50	1069	2136	1324	0.01	0.39904E+02	0.92025E−05
EXTROSNB	1000	3413	6971	3601	0.69	0.30331E−05	0.73937E−05
FLETCBV2	5000	0	1	1	0.01	−0.50027E+00	0.79960E−07
FLETCHCR	1000	6741	14004	7339	2.14	0.46043E−10	0.93221E−05
FREUROTH	5000	65	123	91	0.20	0.60816E+06	0.53153E−05
GENHUMPS	5000	6718	13563	6863	20.37	0.30994E−10	0.28115E−05
GENROSE	500	1267	2564	1298	0.18	0.10000E+01	0.71356E−05
INDEF	5000	1	53	52	0.14	−0.48234E+37	0.29028E+02
LIARWHD	5000	21	48	32	0.04	0.74193E−17	0.35095E−06
MANCINO	100	11	23	12	0.30	0.20215E−17	0.97074E−06
MSQRTALS	1024	2443	4893	2452	8.97	0.27377E−07	0.93392E−05
MSQRTBLS	1024	1907	3820	1914	7.04	0.75276E−08	0.98778E−05
NONCVXU2	5000	7449	14877	7472	14.99	0.11585E+05	0.90280E−05
NONDIA	5000	9	29	23	0.04	0.13578E−16	0.51587E−05
NONDQUAR	5000	2194	4415	2243	1.62	0.29928E−05	0.97204E−05
PENALTY1	1000	43	104	66	0.01	0.96880E−02	0.85578E−05
PENALTY2	200	181	216	329	0.03	0.47116E+14	0.94962E−05
POWELLSG	5000	123	250	148	0.09	0.40020E−05	0.82508E−05
POWER	10000	346	693	347	0.33	0.78877E−08	0.96556E−05
QUARTC	5000	32	65	33	0.02	0.28560E−04	0.30250E−05
SCHMVETT	5000	35	63	46	0.19	−0.14994E+05	0.90275E−05
SENSORS	100	23	55	40	0.19	−0.21085E+04	0.29929E−05
SINQUAD	5000	43	106	92	0.17	−0.67570E+07	0.25394E−05
SPARSQUR	10000	20	41	21	0.15	0.27336E−06	0.67734E−05
SPMSRTLS	4999	186	379	195	0.51	0.19502E−08	0.82452E−05
SROSENBR	5000	12	26	16	0.02	0.28355E−18	0.23712E−09
TESTQUAD	5000	1623	3247	1624	0.76	0.65673E−11	0.95924E−05
TOINTGSS	5000	3	7	4	0.01	0.10002E+02	0.60471E−05
TQUARTIC	5000	20	61	49	0.04	0.29181E−22	0.12818E−08
TRIDIA	5000	738	1477	739	0.49	0.46474E−12	0.94542E−05
VARDIM	200	28	57	29	0.00	0.27052E−22	0.20805E−08
VAREIGVL	50	52	142	90	0.00	0.40508E−10	0.71079E−05
WOODS	4000	155	354	223	0.19	0.13943E−07	0.84174E−05
EG2	1000	3	7	4	0.00	−0.99895E+03	0.81318E−05
FMINSRF2	5625	305	613	308	0.80	0.10000E+01	0.96355E−05
FMINSURF	5625	420	841	421	1.16	0.10000E+01	0.95658E−05
MOREBV	5000	32	65	34	0.04	0.10103E−08	0.89736E−05

Table 1 (*continued*)

Prob	Dim	Iter	Nf	Ng	Time	Fv	Norm
DECONVU	61	102	205	103	0.00	0.37223E−06	0.98843E−05
TOINTGOR	50	107	206	121	0.00	0.13739E+04	0.88894E−05
TOINTQOR	50	28	55	31	0.00	0.11755E+04	0.49970E−05
SPARSINE	5000	16571	33143	16572	50.33	0.27821E−08	0.90710E−05

Table 2
Test results for CGCBB($M = 3$).

Prob	Dim	Iter	Nf	Ng	Time	Fv	Norm
ARGLINA	200	1	3	2	0.00	0.20000E+03	0.77887E−07
ARGLINB	200	22	143	145	0.13	0.99625E+02	0.17497E−02
ARGLINC	200	9	32	31	0.02	0.10113E+03	0.28725E−04
ARWHEAD	5000	9	20	13	0.02	0.00000E+00	0.34409E−05
BDQRTIC	5000	1512	3130	1858	3.58	0.20006E+05	0.88874E−05
BROWNAL	200	3	8	5	0.00	0.14732E−08	0.31408E−05
BROYDN7D	5000	1480	2955	1491	5.27	0.19876E+04	0.61207E−05
BRYBND	5000	26	53	27	0.08	0.20005E−10	0.68958E−05
CHAINWOO	4000	306	607	328	0.59	0.45728E+01	0.95076E−05
CHNROSNB	50	281	564	283	0.00	0.26033E−11	0.88767E−05
COSINE	10000	12	40	38	0.10	−0.99990E+04	0.22317E−05
CRAGGLVY	5000	96	188	123	0.36	0.16882E+04	0.75127E−05
CURLY10	10000	69148	85473	122022	273.74	−0.10032E+07	0.98769E−05
CURLY20	10000	79465	99798	138691	472.46	−0.10032E+07	0.98831E−05
DIXMAANA	3000	8	17	9	0.01	0.10000E+01	0.15239E−05
DIXMAANB	3000	9	19	10	0.01	0.10000E+01	0.42823E−06
DIXMAANC	3000	10	21	11	0.01	0.10000E+01	0.11099E−05
DIXMAAND	3000	12	25	13	0.02	0.10000E+01	0.14429E−05
DIXMAANE	3000	190	381	191	0.16	0.10000E+01	0.98000E−05
DIXMAANF	3000	147	295	148	0.12	0.10000E+01	0.98334E−05
DIXMAANG	3000	139	279	140	0.12	0.10000E+01	0.97285E−05
DIXMAANH	3000	137	275	138	0.12	0.10000E+01	0.84625E−05
DIXMAANI	3000	838	1677	839	0.70	0.10000E+01	0.93634E−05
DIXMAANJ	3000	142	285	143	0.12	0.10000E+01	0.99872E−05
DIXMAANL	3000	108	217	109	0.10	0.10000E+01	0.85858E−05
DIXON3DQ	10000	10000	20001	10002	14.70	0.12228E−11	0.39539E−06
DQDRTIC	5000	7	15	8	0.01	0.56416E−14	0.14985E−06
DQRTIC	5000	24	49	25	0.01	0.26146E+02	0.14037E+00
EDENSCH	2000	32	61	39	0.03	0.12003E+05	0.66234E−05
ENGVAL1	5000	25	51	32	0.05	0.55487E+04	0.60756E−05
ERRINROS	50	660	1316	832	0.00	0.39904E+02	0.71000E−05
EXTROSNB	1000	2404	4977	2619	0.50	0.37044E−05	0.80336E−05
FLETCBV2	5000	0	1	1	0.00	−0.50027E+00	0.79960E−07
FLETCHCR	1000	7650	16693	9208	2.49	0.50585E−10	0.93486E−05
FREUROTH	5000	153	202	277	0.52	0.60816E+06	0.77104E−05
GENHUMPS	5000	7522	15140	7651	22.10	0.51403E−09	0.56841E−05
GENROSE	500	1424	2882	1467	0.21	0.10000E+01	0.87412E−05
INDEF	5000	1	53	52	0.13	−0.48234E+37	0.71613E+01
LIARWHD	5000	21	46	28	0.04	0.10736E−17	0.55258E−07
MANCINO	100	11	23	12	0.29	0.19921E−17	0.93903E−06
MSQRTALS	1024	2370	4747	2379	8.61	0.43961E−07	0.98424E−05
MSQRTBLS	1024	1785	3576	1793	6.61	0.14021E−07	0.94471E−05
NONCVXU2	5000	8797	16155	10238	18.78	0.11584E+05	0.90684E−05
NONDIA	5000	7	18	12	0.02	0.34283E−15	0.30752E−05
NONDQUAR	5000	1148	2302	1172	0.85	0.91253E−05	0.80675E−05
PENALTY1	1000	38	87	51	0.01	0.96920E−02	0.70527E−05
PENALTY2	200	181	216	329	0.03	0.47116E+14	0.99175E−05
POWELLSG	5000	175	359	209	0.13	0.63782E−04	0.99607E−05
POWER	10000	385	771	386	0.38	0.37728E−07	0.90099E−05
QUARTC	5000	24	49	25	0.01	0.26146E+02	0.14037E+00
SCHMVETT	5000	34	61	43	0.18	−0.14994E+05	0.78930E−05
SENSORS	100	25	58	37	0.18	−0.21060E+04	0.85451E−05
SINQUAD	5000	201	332	488	0.80	−0.67570E+07	0.60589E−05
SPARSQUR	10000	25	51	26	0.18	0.58856E−06	0.92330E−05
SPMSRTLS	4999	174	355	183	0.47	0.26023E−08	0.86490E−05
SROSENBR	5000	11	24	16	0.01	0.43301E−08	0.84650E−05
TESTQUAD	5000	1569	3139	1570	0.74	0.24372E−10	0.94727E−05
TOINTGSS	5000	3	7	4	0.01	0.10002E+02	0.60473E−05

(*continued on next page*)

Table 2 (*continued*)

Prob	Dim	Iter	Nf	Ng	Time	Fv	Norm
TQUARTIC	5000	15	42	32	0.03	0.94463E−12	0.12276E−05
TRIDIA	5000	736	1473	737	0.47	0.55078E−12	0.98780E−05
VARDIM	200	27	55	28	0.00	0.45298E−12	0.26921E−03
VAREIGVL	50	15	39	24	0.00	0.25171E+00	0.44808E+00
WOODS	4000	153	366	229	0.19	0.16376E−07	0.72184E−05
EG2	1000	3	7	4	0.00	−0.99895E+03	0.77848E−05
FMINSRF2	5625	294	592	298	0.77	0.10000E+01	0.99148E−05
FMINSURF	5625	426	854	428	1.17	0.10000E+01	0.90216E−05
MOREBV	5000	36	73	38	0.04	0.81803E−09	0.89526E−05
DECONVU	61	124	249	125	0.00	0.38374E−06	0.82500E−05
TOINTGOR	50	105	203	122	0.00	0.13739E+04	0.94533E−05
TOINTQOR	50	27	54	29	0.00	0.11755E+04	0.70083E−05
SPARSINE	5000	16187	32375	16188	47.42	0.27185E−08	0.93852E−05

a winner. It clear see that CGCBB is superior to CG_DESCENT at least 15%. By taking everything all in all, we conclude that our proposed method provides an efficient approach to solve large-scale unconstrained optimization problems and performs much better than CG_DESCENT.

Acknowledgments

We would like to thank the anonymous referees for their useful comments and suggestions which improved this paper. The first author's work is supported by Chinese NSF grant 11001075, and the Natural Science Foundation of Henan Province Education Department grant 2010B110004.

References

[1] J. Nocedal, J.S. Wright, Numerical Optimization, Springer Verlag, New York, Inc., 1999.
[2] R. Fletcher, C. Reeves, Function minimization by conjugate gradients, The Comput. J. 7 (1964) 149–154.
[3] E. Polak, G. Ribière, Note sur la convergence de directions conjugées, Rev. Fr. Inform. Rech. Oper. 16 (1969) 35–43. 1969.
[4] M.R. Hestenes, E.L. Stiefel, Methods of conjugate gradient for solving linear systems, J. Res. Nat. Bur. Stand. 49 (1952) 409–436.
[5] Y.H. Dai, Y. Yuan, A nonlinear conjugate gradient method with a strong global convergence property, SIAM J. Optim. 10 (1999) 177–182.
[6] M.J.D. Powell, Nonconvex minimization calculations and the conjugate gradient method, in: Numerical Analysis, Dundee, 1983, in: Lecture Notes in Mathematics, vol. 1066, Springer, Berlin, Germany, 1984, pp. 122–141.
[7] W.W. Hager, H. Zhang, A survey of nonlinear conjugate gradient methods, Pac. J. Optim. 2 (2006) 35–58.
[8] J.C. Gilbert, J. Nocedal, Global convergence properties of conjugate gradient methods for optimization, SIAM J. Optim. 2 (1992) 21–42.
[9] L. Grippo, S. Lucidi, A globally convergent version of the Polak–Ribière conjugate gradient method, Math. Program. 78 (1997) 375–391.
[10] G. Li, C. Tang, Z. Wei, New conjugacy condition and related new conjugate gradient methods for unconstrained optimization, J. Comput. Appl. Math. 202 (2007) 523–539.
[11] Y.H. Dai, L.Z. Liao, New conjugate conditions and related nonlinear conjugate gradient methods, Appl. Math. Optim. 43 (2001) 87–101.
[12] W.W. Hager, H. Zhang, A new conjugate gradient method with guaranteed descent and an effcient line search, SIAM J. Optim. 16 (2005) 170–192.
[13] J. Zhang, Y. Xiao, Z. Wei, Nonlinear conjugate gradient methods with sufficient descent condition for large-scale unconstrained pptimization, Math. Prob. Eng., Volume 2009, Article ID 243290, 16 pages.
[14] X.M. An, D.H. Li, Y. Xiao, Sufficient descent directions in unconstrained optimization, Comput. Optim. Appl. 48 (2011) 515–532.
[15] J. Barzilai, J.M. Borwein, Two point step size gradient method, IMA J. Numer. Anal. 8 (1988) 141–148.
[16] A.R. Conn, N.I.M. Gould, Ph.L. Toint, CUTE: constrained and unconstrained testing environment, ACM T. Math. Software 21 (1995) 123–160.
[17] Y.H. Dai, L.Z. Liao, *R*-linear convergence of the Barzilai and Borwein gradient method, IMA J. Numer. Anal. 26 (2002) 1–10.
[18] M. Raydan, On the Barzilai and Borwein choice of steplength for the gradient method, IMA J. Numer. Anal. 13 (1993) 321–326.
[19] M. Raydan, The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem, SIAM J. Optim. 7 (1997) 26–33.
[20] E.G. Birgin, J.M. Martínez, M. Raydan, Nonmonotone spectral projected gradient methods on convex sets, SIAM J. Optim. 10 (2000) 1196–1211.
[21] Y. Xiao, Q. Hu, Subspace Barzilai–Borwein gradient method for large-scale bound constrained optimization, Appl. Math. Optim. 58 (2008) 275–290.
[22] Y.H. Dai, W.W. Hager, K. Schitkowski, H.C. Zhang, The cyclic Barzilai–Borwein method for unconstrained optimization, IMA J. Numer. Anal. 26 (2006) 1–24.
[23] Y.H. Dai, On the asymptotic behaviour of some new gradient methods, Math. Program. 103 (2005) 541–559.
[24] G. Zoutendijk, Nonlinear programming, computational methods, in: J. Abadie (Ed.), Integer and Nonlinear Programming, North-holland, Amsterdam, 1970, pp. 37–86.
[25] W.W. Hager, H. Zhang, Algorithm 851: CG_DESCENT, a conjugate gradient method with guaranteed descent, ACM T. Math. Software 32 (2006) 113–137.
[26] E.D. Dolan, J.J. Moré, Benchmarking optimization software with performance profiles, Math. Program. 91 (2002) 201–213.